Editorial Comment ■

Service-oriented Architecture in Medical Software: Promises and Perils

PRAKASH M. NADKARNI, MD, RANDOLPH A. MILLER, MD

■ **J Am Med Inform Assoc.** 2007;14:244–246. DOI 10.1197/jamia.M2349.

In the current issue of *JAMIA*, Kawamoto and Lobach[1] propose a software framework intended to facilitate widespread, effective, clinical decision support. The proposed framework embraces a service-oriented-architecture (SOA) approach. Service-oriented-architecture is a philosophy of design described as "the software equivalent of Lego bricks,"[2] where a toolset of mix-and-match units ("services"), each performing a well-defined task, can reside on different machines (including geographically separated ones), ready to be used when needed. The most widespread implementations of SOA involve the use of Web services, where a given computational resource/service can be invoked by a remote machine via messages composed in XML and sent over HTTP, so that they can operate across firewalls. Think of a Web service, in its simplest form, as a subroutine that can be called over the Internet.

Thanks to success stories such as Amazon.com,[3] and at software giants such as SAP,[4,5] the status of SOA in the business information technology (IT) domain has risen meteorically. The Kawamoto and Lobach paper[1] reiterates the following potential benefits of SOA:

- Simpler software design and implementation, by decomposing complex problems into smaller, more manageable ones.
- Improved software reusability through enhanced reuse of existing IT resources.
- Improved adaptability to changing business requirements.
- Cost savings consequent to the above benefits.

Some IT articles, however, present a "caveat emptor" skepticism about SOA.[6,7] One must carefully inspect the claimed benefits of SOA closely, examine its potential drawbacks, and take a balanced approach to any proposed new SOA application while taking into consideration past lessons from business IT.

Affiliations of the authors: Yale Center for Medical Informatics, Yale University School of Medicine (PMN), New Haven, CT, and Department of Biomedical Informatics, Vanderbilt University (RAM), Nashville, TN.

Correspondence and reprints: Randolph A. Miller, MD, Eskind Biomedical Library-Lower Level, 2209 Garland Avenue, Nashville, TN 37232-8340; e-mail: <randolph.a.miller@vanderbilt.edu>.

SOA Benefits Analyzed

In software design, simplification through problem decomposition into semi-independent units (subroutines, classes) is an important, incontrovertible principle. In SOA, where the units (services) lead a relatively autonomous existence (e.g., units are free to reside on physically separate hardware), decomposing a software solution into individual services makes sense only in two circumstances:

- When a particular service can provide independent value to the remote caller.
- When physical separation of services on multiple machines provides quantifiable scalability and performance benefits through hardware parallelism that more than offset the extra overhead of intermachine communication. Typically, the latter economies occur when the services are used mostly internally, and even then, achieving scalability may require upgrades to internal network infrastructure to minimize this overhead. When a user of Amazon.com asks for information on a book, for example, several dozen independent services compose individual parts of the returned Web page, such as product information, book rank, similar titles, and customer reviews.

Newman[7] points out that reuse of an existing resource requires that the resource was designed to be reusable. Designing, or redesigning, for reusability involves one or more of the following steps:

- Rigorously examining the assumptions implicit in an existing resource's design and determining which of these do not hold in a different circumstance. For example, consider the situation where a service's output must eventually be converted into human-readable text, as with Amazon's Web pages, or the recommendation system for preventive clinical procedures postulated in this issue's paper. One issue to be determined is whether the service should have a transnational reach, so that output is not restricted to the English language alone. Creating localizable software (i.e., software that can be adapted for different cultures without having to completely reprogram it) involves, among other things, isolating the textual elements that are to be shown to the user in "resource files" and referring to them symbolically. While software tools to facilitate localization are quite mature (e.g., Microsoft Visual Studio Team Suite®), re-

working an existing software resource to make it localizable is inherently a painstaking and error-prone task.
- Identifying a general underlying problem amenable to creation of a general solution.
- Deciding to make something reusable often requires changing an organization's business model. Linquist[2] cites ProCard, a credit-card service provider whose strengths resided in its software suite. ProCard determined that converting its suite into independently accessible services was worthwhile only if it transformed itself from a competitor of Visa and MasterCard to a provider of software services to its much larger rivals.
- Deciding whether the extra effort needed to re-engineer and/or generalize the solution is justified in the present situation, given constraints such as urgency and limited developer resources.

The caveats here are that, as in great architecture and literature, simplicity is an elusive entity whose achievement is as much art as science: that is why it is so uncommon. Regarding reusability, Newman[7] estimates that it generally takes at least three iterations to design a reusable software unit (class, library, service) well. Individuals capable of abstracting problems and devising elegant, simple, and general solutions tend to be both rare and expensive.

It might be more accurate to state that SOA is a strategic, long-term objective rather than a tactical, short-term one. As with other system design approaches—e.g., metadata-driven software architectures—developers must invest a significant amount of effort in framework-building, which takes time, resources, and human expertise, before anticipated payoffs in reduced costs and greater adaptability are realized. For both Amazon and SAP, the time investment was at least four years.

The Issue of Standards

For a service that can provide independent value, the need to adhere to existing standards (when the service may be called from outside the organization, and must be interoperable with services designed by others) often necessitates extra effort that would not otherwise be required. Here the challenges concern the semantics of the data that must be passed to and from the service and how it must be represented, rather than the nature of the XML plumbing: modern software development environments mostly shield you from the latter and allow you to concentrate on the former.

While the paper by Kawamoto and Lobach emphasizes the use of a standard, the HL7 Service Functional Model (SFM) Specification for Decision Support Service[8] we note that this standard is in its early stages, with version 0.85 posted June 19, 2006 and version 1.0 posted July 23, 2006. In addition, we note that Dr. Kawamoto is a member of the HL7 SOA SIG, and was project leader on the HL7 Decision Support Service effort. Considerable experience will be needed to determine whether the specification is sufficiently comprehensive to meet its objectives and to what extent it will need to be revised.

Amazon was fortunate in that they were the sole standards-setter for the family of services that they offer for external use. In an area where consensus is not yet present, however, the lead time required for consensus to be achieved must be factored in. In the medical domain, where HL7 version 3's status is still not official (for reasons that are not entirely technical but beyond the scope of this editorial), the reasons and process for implementing SOA must be similar to the ones in IT elsewhere. That is:

- The case for SOA must be made from a business perspective.
- The initial implementations of SOA must focus on internal use (where the services are used primarily over the organization's Intranet) so that one can worry less about nascent and immature standards and focus on solving specific problems.
- SOA must be understood to be a long-term, strategic approach that is not always applicable. In particular, as stated in Lindquist,[2] if the reusability quotient of a given problem is low, a "one-off" non-SOA solution turns out to be more direct and faster to create.

Service Discovery: Theory and Reality

The paper by Kawamoto and Lobach[1] mentions "service discovery" as one of the aspects of SOA. The idea of service discovery is that a service contains descriptive information about itself that would allow a remote automated software agent to determine whether a service exists on the Internet that would address a particular need. The HL7 specification discussed above provides for keywords that would assist the search for a particular service.

While "semantic Web" researchers have made much of the potential of service discovery, this is currently far removed from reality. When carefully considered, service discovery is an extremely difficult problem to solve. If we regard the collection of biomedical services as a vast Internet-accessible subroutine library, determining whether a service is appropriate for a specific task would require considerable expert human intervention. Software developers who have to work with giant development frameworks such as Java® and Microsoft.NET®, which contain tens of thousands of subroutines, have to deal regularly with the same problem in a non–Web services setting. As often as not, browsing vendor documentation is insufficient, forcing the developer to use a Web search engine such as Google® and browse resources such as developer group forums and blogs.

At present, the large Web service providers simply provide extensive documentation on each service, which also includes programming examples as well as case histories of successful use. The best service providers continually solicit feedback on the quality of their documentation in order to improve it.

The Issue of Medical Errors and Adverse Effects

Clinicians, health care organizations, consumer groups, and informatics developers regard clinical decision support as a key method to address the inefficiencies and errors documented to occur in busy clinical practices.[9,10] However, many health care organizations now see decision support as a selective means to provide better care to their patients in a manner that distinguishes them from their competitors. For the reasons listed above, SOA within a local environment may greatly facilitate such clinical decision support. However, there are potentially severe concerns if an organization "outsources" its decision support to potentially imperfect external agencies.

The manner in which SOA is implemented, as noted by Kawamoto and Lobach, is as a series of "black boxes" that, given an input, produce an output. In general, SOA services are not "licensed practitioners," so that legally, the patient's health care provider (clinician or institution) is responsible for overriding any erroneous advice provided by an SOA service. The latter circumstances may in and of themselves inhibit reliance on external clinical SOA services.

If one were to ask a commercial software vendor, a pharmaceutical company, or a health care-related agency to provide a "medication dosing service" at a regional or national level, who would trust the service to provide uniformly correct answers? If a hypothetical SOA medication dosing service only took age into consideration, and not weight, it could not be used in pediatrics or for dosing certain medications, such as aminoglycoside antibiotics, in adults. Even if a dosing service took weight and age into consideration, but not gestational age or height, it still could not be used for neonatal applications or for situations in which doses are based on body surface area (e.g., chemotherapy). Dosing also depends on renal and hepatic function. Finally, dosing is often diagnosis-dependent. The same adult patient, at a given age, weight, height, and level of renal function, would require far higher doses of a "correct" beta lactam antibiotic when treating bacterial endocarditis in an inpatient setting than would be required to treat that patient's community-acquired pneumonia in an outpatient setting. How much information would a "medication dosing service" require as input in order to give proper advice in all settings, and how complex would the output have to be to cover a myriad of potential patient characteristics? In the best commercial drug databases on the market today, the algorithmic and data-structure sophistication for such a dosing service does not yet exist; the best that these systems do is reproduce the medication vendor's package insert text. Converting such an existing imperfect solution to a Web service would not be satisfactory.

The logistics of responsibly implementing long-distance clinical SOA services may be overwhelming. The "black box" model for SOA services assumes relatively minimal input and output. Consider, however, the circumstance whereby the maintainers of a regional or national SOA clinical decision support service discover that the advice provided by the service has been faulty over the past 24 hours due to a "bug" introduced into the software (or the knowledge base) of the service. What mechanisms would exist that would enable the SOA provider to contact all care providers who had relied upon the service to provide advice for patients, and to then determine which patients' ordered regimens would need to be changed? To do so would require an "SOA transaction identifier" that at least goes back to identifying the institution requesting the service, and possibly even to the level of identifying the patient (at the SOA level). This raises the possibility whereby enough information about a patient must be transmitted to an SOA service to obtain advice that HIPAA privacy rules are violated.

The Issue of Commercialization

If SOA services are vended commercially (as might be the case in some future scenario involving the authors' commercialization efforts), one must consider how to charge the users of the services. Some of Amazon's services, e.g., product data (prices, images, customer reviews), are free, while other services are priced: e.g., the Historical Pricing service, which provides access to more than three years of Amazon's sales data for any item, has a fee of $249/month for up to 60,000 requests per month. The legal and ethical issues related to how such a commercial SOA decision support system might operate across state, and possibly national borders, have yet to be addressed in a definitive and thoughtful manner.

References ■

1. Kawamoto K, Lobach D. Proposal for Fulfilling Strategic Objectives of the U.S. Roadmap for National Action on Decision Support through a Service-Oriented Architecture Leveraging HL7 Services. J Am Med Inform Assoc. 2007;14:146–155.
2. Lindquist C. A New Blueprint for IT (editorial). CIO Magazine 2005. Available at: http://www.cio.com/archive/081505/soa.html. Accessed December 6, 2006.
3. Gray J. A conversation with Werner Vogels, CTO, Amazon.com. Web Services 2006. Available at: http://portal.acm.org/ft_gateway.cfm?id=1142065&type=pdf. Accessed December 6, 2006.
4. SAP. SAP - Enterprise Service-Oriented Architecture+. 2006; Available at: http://www.sap.com/platform/esa/index.epx. Accessed December 2, 2006.
5. Campbell S, Mohun V. Mastering Enterprise SOA with SAP NetWeaver and mySAP ERP. New York: Wiley; 2006.
6. Bakker B. The reality of SOA Caveat Emptor. iWeek 2006 Aug 31, 2006 Available at: http://www.iweek.co.za/ViewStory.asp?StoryID=165822. Accessed December 2, 2006.
7. Newman J. SOA, Reuse, Caveat Emptor. 2004; Available at: http://integralpath.blogs.com/thinkingoutloud/2004/11/soa_reuse_cavea.html. Accessed December 5, 2006
8. Kawamoto K, Esler B. Clinical Decision Support TC and Service Oriented Architecture SIG: Service Functional Model Specification, Decision Support Services: Version 1.3, June 19, 2006. 2006. Available at: http://hssp-dss.wikispaces.com/space/showimage/HL7+Decision+Support+Service+(DSS)+Service+Functional+Model+(SFM),+v0.85.doc. Accessed December 10, 2006.
9. Kohn LT, Corrigan JM, Donaldson M, eds. *To Err Is Human: Building a Safer Health System.* Washington, DC: Institute of Medicine; 1999.
10. The Leapfrog Group. *Computer Physician Order Entry.* 2006. Available at http://www.leapfroggroup.org/for_hospitals/leapfrog_safety_practices/cpoe. Accessed December 2, 2006.